SRPeek: Super Resolution Enabled Screen Peeking

***Abstract*— We use smartphones everywhere and privacy concerns have arisen for the "shoulder surfing" attack, where strangers peek at our phone in public areas. To mitigate this privacy threat, various countermeasures have been designed but mainly against attackers with the naked eye. With the development of smartphones and super resolution (SR) technique, a malicious attacker can peek from farther away with the assist of his/her smartphone camera and SR algorithms, rendering the underestimated threat model. To underline this concern, we present SRPeek, an end-to-end system of shoulder surfing deployed on commercial smartphones, including a unique deep neural network (DNN) architecture for multi-image SR. We implement SPPeek in Android. The experimental results demonstrate its efficiency and robustness, allowing a human to read 95% text contents when the distance between the camera and screen is as long as 1.5m in different scenarios. And it outperforms the state-of-the-arts and fills the blank space of multi-image SR for shoulder surfing attack.**

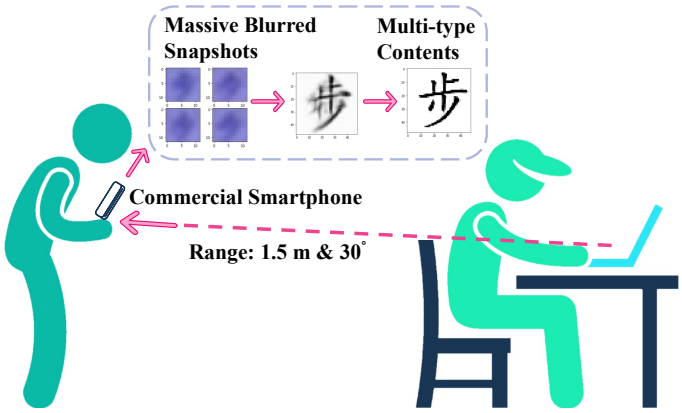


Fig. 1: Illustration of the long-range shoulder surfing threat model on smartphones, exacerbated by the SR algorithm.

I. Introduction

Smartphones have become a necessity in our lives, as we are checking our mobile phones constantly throughout the day. As a result, some privacy concerns have arisen about nearby parties peeking at our screen, namely "shoulder surfing". Given that the attacker is not malicious and merely takes several peeks out of curiosity, most works in this area are built on a threat model of an attacker observing with naked eyes, avoiding the severe data leakage effectively [6]. For example, a mere stranger can hardly do any harm reading a fragment of the correspondence or acquiring the password shown on screen without any equipment.

The privacy threat however has been exacerbated with the rapidly developed mobile camera module these years. Equipped with multiple cameras, the newest generation of smartphones can perform 100× zooming compared to the standard 5× of single camera phones. Hardware improvements in memory also allows the burst mode at tremendous frame rates and even high-speed photography, delivering videos with thousands of frames per second. And more images mean more recorded information of the victim. Given the commercial smartphone, the attacker can record the information reliably for propagating. For example, in a Senate hearing, the Justice Secretary of Philippines, Vitaliano Aguirre II, suffered a leakage of his text messages, as someone had taken a snapshot of his smartphone [20]. To gain vital data more stealthily and accurately from even further away, the processing technique of multi-frame super resolution (SR) algorithms can be further employed by the attacker, delivering a long-range shoulder surfing threat model.

Unfortunately, none of existing work can mitigate the new threat model. Specifically, massive efforts have been made, from physical privacy films to alternative password entry interfaces [19], [25] and input methods [13]. All of these methods however require additional deployment cost [3] and cannot be widely deployed for critical privacy stages (e.g., password entry) of most scenarios. Furthermore, none of existing works can deal with the presence of developed smartphone cameras and SR algorithms in shoulder surfing scenarios. Given the ubiquitous usages of smartphones and serious damages of the data disclosure for the victim, it is imminent to recall attention on this new privacy threat and propose the corresponding countermeasure for modern circumstances.

Threat Model. We present SRPeek, an end-to-end system of shoulder surfing using the SR algorithm, illustrated in Figure 1. Assuming the victim is scanning messages in life, the attacker can raise his commercial smartphone with powerful lenses and our SR network to the victim stealthily, even from a long range (e.g., 1.5 meters far away within the visual angle of 30°). As the camera zooms in, he can take 20 to 30 snapshots continuously in burst mode, with about 0.05 0.1 seconds of interval between frames. This process can take about 2 seconds with the consistent content and position for the screen of the victim. Benefited from our multi-frame SR neural network, those blurred snapshots can be then aligned, cropped, and processed, delivering a holistic threat model for acquisition of high-resolution contents of the victim in real time. Note that multi-type contents can be recovered accurately on the smartphone screen of the victim, covering numbers, English and Chinese characters. The above procedure can be automated by an app and repeated every few seconds, giving the attacker a continuous surveillance over the victim's screen.

Challenges. Creating a high-resolution shoulder surfing threat

model, however, is not trivial, especially for commercial smartphones in real time. And we achieve SRPeek by overcoming four key challenges.

- **Blurriness of input images.** The most prominent one is the blurriness of captured snapshots, as they are taken secretly and hurriedly at extreme range, without specially focusing. It can be further exacerbated by straining the magnification of the smartphone lenses. Unlike most SR applications and datasets working with "normally" captured snapshots (e.g., scenery, scanned images) [16], [18], these snapshots exhibit lower concentrations of information and more artifacts, requiring a reliable "reconstruction" than "interpolation". Since blurrier input means less information, rendering a greater possibility of reconstructing the wrong character. On one hand, each snapshot is blurred with a randomly different PSF kernel, removing the consistency between neighboring frames CL: (citation?), shown in Figure 1. Thus it cannot be approximated as video clips using a constant, isotropic gaussian kernel; on the other hand, each defocused snapshot only contains fractions of information, analogous to pieces of a jigsaw puzzle. Few useful information can be extracted separately and merged directly to produce perceptible results if processed by common procedures to enhance the resolution of blurred snapshots CL: (citation?).
- **Super resolution with multiple frames.** To resolve the blurriness of input images, a SR algorithm on a smartphone is required to recover the captured contents. However, none of existing SR algorithms can be adapted for this attack scenario, especially with massive extremely blurred snapshots as input. Since smartphones can capture 10 snapshots per second in burst mode. Specifically, the difference between videos and multiple snapshots rules out existing image/video super resolution algorithms CL: (citation?). The reason has two folds. First, either increased input leads to increases in model complexity, making it unsuitable for mobile deployment. Second, the nature of our task requires to do pixel-wise correlation and comparison with same coordinates on consecutive images throughout our construction process. Otherwise we cannot tell if, for example, it is one stroke or two parallel strokes that is shown in these two darker pixels on this snapshot.
- **Model complexity and variable input.** We have to operate our threat model locally on commercial smartphones in a real-time manner rather than remotely in cloud server. Since 1) the latter would require stable Internet access, and 2) sending multiple images across the Internet will consume time and bandwidth. Given the required 10 to 20 images to process for ideal content recovery, we have 1-2s as the minimum interval between each output frame, rendering limited period for calculation with restricted power and RAM. In cases where the display on the target screen changes more frequently, less images and processing time are available for each scene. Consequently, it cannot only require a smaller and simpler network model, but also a model that can deal with dynamic inputs efficiently, either 3 or 20 images.
- **Complexities of characters.** To deploy our threat model broadly for real-life scenarios (e.g., record passwords or e-mails), it is supposed to reconstruct multi-type texts, including Chinese characters, English letters, and numbers. Composed of multiple strokes, characters are distributed discretely in the pixel-wise image and different with other entities (e.g., faces and natural objects), rendering unique challenges. In some cases, messing up a stroke even slightly can shorten or lengthen a stroke, leading to a different character for misunderstandings. Unlike most SR applications working on similarity between their output and the ground truth, our network architecture and training process must be engineered to recover readable contents. Another challenge is the imbalanced element for words. Since we know certain stroke or alphabet of Chinese and English characters can occur more frequently, results in lower prediction accuracy over the less common, unconventionality shaped characters.

Summary of Evaluation Results. Equipped with our uniquely designed SR algorithm, our system can help the attacker read the characters shown on the victim's phone with above 90% accuracy at 2m distance on a smartphone without optical zooming, or 6m distance on a phone with optical zooming in our experimental settings. The system is able to capture snapshots and produce high-res images constantly with about 2 seconds interval. Our experiments with real life scenarios show that this system is able to decipher texts or passwords of the victim at a safe distance, without alarming the victim, posing a threat to screen privacy.

Contributions. This paper makes the following contributions:

- We propose SRPeek, an end-to-end threat model of shoulder surfing from a broad range. To the best of our knowledge, we are the first to consider the presence of smartphone cameras and SR algorithms in shoulder surfing scenarios.
- We design a multi-frame SR neural network architecture aiming at reconstructing extremely massive blurred and defocused snapshots. It can be further extended to other text-recognition applications on commercial smartphones.
- We evaluate this new shoulder surfing attack in comprehensive scenarios. And it outperforms the state-of-the-arts for the new privacy concern.

The rest of the paper is organized as follows. Section II describes the related work, especially the state-of-the-arts for the shoulder surfing and SR techniques. We present the system and network design in Section IV, followed by the implementation and evaluation in Section V and VI. We further wrap up this paper by delivering the limitations and countermeasures of our system in Section VII. And the conclusion is shown in Section VIII.

II. Related Work

A. Shoulder Surfing

With the arrival of the information era, privacy issues are becoming increasingly prominent. Smartphone screen privacy, the concern of our smartphones being observed by strangers

TABLE I: A comparison of state-of-the-art works on shoulder surfing.

Reference	Deployment	Range[a]	Subject	Performance
Wision [?]	Intensity	USRP-N210	SAR	(1,8×8)
RF_Avatar [?]	Intensity	FMCW Radio	DNN	(4,16)
C. Karanam [?]	Intensity (RSSI)	COTS Wi-Fi	SAR	(1,150×150)[c]
P. Proffitt [?]	Intensity	USRP-N210	SAR & DNN [?]	(180, 180)[b]
P. Holl [?]	Intensity & Phase	3D Hologram Simulation	Wave Front [?]	(1,50×40)[d]
SRPeek	COTS smartphones	1.5m & 30°	Multi-type Characters	Real-time Monitoring with 95% Contents

[a]The maximum distance and observing view angle to the victim's screen.

in public areas, or shoulder surfing, have been studied heavily recently [6], [9], [14]. To mitigate this threat, some systems hide the information [1] or warn the user [21] once sensing malicious passers-by; others modify the user interfaces, including creating honeypots (for passwords) [2], confusing unauthorized parties [25], and making the interactions invisible [13] or unreadable from a distance [3]. Most of the works assume that the attacker is a casual passer-by, taking occasional peeks with the naked eye, as is the case most of the time [6] CL: (more citation and table 1 row-1). Given the assisted equipment, the malicious attacker can however acquire the sensitive information (passwords, business correspondence, etc.) readily to do real harm. To deal with the threat model of tool-assisted shoulder surfing CL: (more citation and table 1 row-2), Maggi et al. designed an automatic shoulder surfing threat model, observing the target smartphone with a camera [17]. It can however only function when the attacker is standing at close range, which is a barely practical scenario. By tailoring and fusing the SR technology with smartphones with developed mobile camera module and processing ability, we propose a stronger shoulder surfing threat model in which attackers can deploy it on commercial smartphones while obtaining information for a longer range to reduce suspicion, say 1.5m away from the victim's screen with an observing view of 30° shown in Table I. Evaluations demonstrate its feasibility and privacy concerns in our daily life.

B. Super Resolution

Image SR is the process of reconstructing an image with a higher spatial resolution. Based on the structural pattern, self-similarity [24], or previous knowledge of the image genre, the single-image SR techniques take as input a single low-resolution image, rendering a sharp, high-resolution one by deducing missing information and reconstructing the missing pixels. Further, multi-image SR techniques work on a set of pictures on the same scene, such as multiple snapshots captured by a smartphone, successive images from a satellite, or adjacent frames on a video clip. These algorithms collect extra data from slight differences of these redundant images, often exhibiting better performance than single-image SR algorithms. And a variety of techniques have been proposed for the multi-image SR problem. And early works focus on the analysis of images in spatial or frequency domains. For example, the Shift-Add algorithm [7] analyzes the spatial information by maximizing the pixel-wise possibility of high-resolution image from low-resolution images. Besides, approaches for the frequency domain rely on the Fourier transform of images. Assuming that the high-resolution scene is band-limited, these algorithms reconstruct high-frequency components from patterns of low-frequency components, removing bands of noise and blur effects simultaneously. Unfortunately, all SR techniques however, are subject-specific and achieve the best performance only for some contents, including natural photos, scanned text, satellite imaging, biometrics CL: (more citation and table 1 row-3). While SRPeek can recover multi-type characters with complex strokes, including Chinese, English and numbers.

Deep learning approaches can also be adopted for Super Resolution. For example, SRCNN [4] was designed using CNN by replacing pooling layers with upsampling layers, achieving notably better performance than previous approaches. It however only focuses on single image SR, as the neural network accepts a single image as input CL: (table 1 row-4). Besides, Generative Adversarial Networks (GAN) were also used to generate photo-realistic images [15], with less artifacts and more genuine-looking details perceptible to the human eye. To resolve multi-image SR tasks, say video SR, existing works [11], [23] design a structure similar to SRCNN, and accept complete video clips as input. Then the sequentiality and consistency between adjacent frames can help recover the missing pixels of clips using convolution networks, Specifically, we can modify the dataflow to merge neighboring frames among the network layers [10], or recurrently processing the frames under the guidance of the output of the previous frame [22] CL: (table 1 row-5).. For images without consistency or sequential information, like satellite images, most works choose hybrid methods to solve the multi-image SR problem with multiple single-image SR procedures. They either merge the results of single-image SR algorithms for efficiency [12], or build a multi-image network to create a comprehensive view based on single image SR networks [5] CL: (table 1 row-6). Due of the extreme blurriness of snapshots, these methods can not deal with the new shoulder surfing threat model we proposed, which takes as input massive blurred snapshots without the consistency between frames. And they achieve limited performance, shown in Table I.

III. System Overview

We implemented a holistic system for shoulder surfing, with the neural network as core, on a smartphone to verify the efficiency of our model. It iterates through the following steps:

Input: Under guidance of the attacker, The smartphone will zoom in, take focus, and take 20 images with burst mode of the target screen. Note that the zoom in step is only for easier interaction and focusing, and to utilize the telephoto lenses and optical zoom, if avaliable, as digital zooming do not put in any extra data; On traditional phones with only digital zooming, the photo-taking will be with 1x zoom, and on phones with optical zooming, 5x to 10x zoom, depending on the optical zooming range. This way information on the images will be more compact and easier to comprehend by neural networks.

Alignment: The images are then aligned to mitigate the shifts between frames caused by hand tremor or movements from the lenses, as cameras with optical lenses will ocationally shift slightly across time due to the movements of its inner mechanism. Luckily, in our scenario the target is a glowing screen whose edges are easily distinguishable in most cases, and we use them as reference to align our images. The images are also spun to make the text horizontal in the process. The screen is cropped out and the rest of the image abandoned.

Adjustment: The lines of text(or stuff differing from the background color of the screen) will be carved out for processing, to reduce the workload of the network. The characters are normally around 10x10 pixels in the photos, and the carved segments will leave 2 pixels of padding on all sides to avoid mutilating the character; A certain amount of error is allowed, but if the size of the text is too small or too large, the neural network will not be able to extract features normally, so the images will be zoomed to the right size(and the zooming of step (1) will be adjusted accordingly).

Processing: As our network accepts only 9x9 patches, we will process all the 9x9 patches among the input photo and merge the results together. After carving a 9x9 patch at corresponding locations of each image, these patches are then processed by our multi-frame super resolution network, generating a single 36x36 image (4x upscaling). The input is RGB colored, while the output, as we are not interested in color, is black and white. When all patches are processed, their outputs are merged together. The overlapping pixels are thus processed: among all the outputs containing this pixel, we collect these values, remove outliers, and use their average as the result. In this way, we generate upscaled image segments of all the characters on the target screen, which are then inserted in one of the input images(roughly upscaled to encompass them, just for reference) and displayed to the attacker.

These steps are repeatedly executed to enable the attacker to monitor the victim at an interval of a few seconds. At critical times requiring continuous monitoring so as not to miss transient display, e.g. password entering, the system can simply lengthen the input phase across that period and process the data afterwards. The workflow of our system is shown in Figure 2.

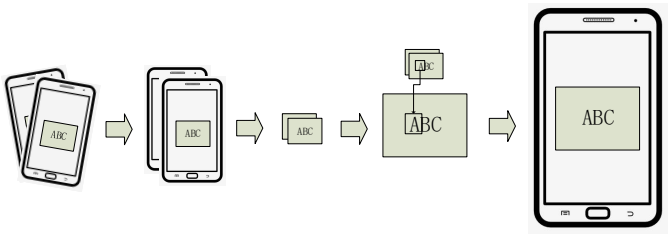


Fig. 2: Illustration of the workflow of our shoulder-surfing system.

IV. System Design

A. Design Principal

To solve the challenges and outperform the state-of-the-arts for this new shoulder surfing attack, we propose a holistic system with the multi-frame SR neural network, illustrated in Figure 2. CL: (These four parts are too long, try to whittle it away to half its length. Refer to the revised contents for challenges in Section introduction.)

- **Layered architecture and frequent introduction of input.** To counter the blurriness of the images, and the tendency of reconstructing fake characters, our network is constructed with several identical SR layers, connected sequentially, and the images are refined step by step throughout the layers. The input and output of each layer is also correspondent to each avaliable image CL: (???), as all input images are processed simultaneously and separately(with the information from other images for reference) throughout the network, all the way to the end of the model where we merge all the data into one output image. Another importent feature of our architecture is that the initial input images are introduced into the dataflow at each layer, from beginning to end, resulting in the uneven depth of our model, acting as an anchor for the reconstruction process so that the output will be faithful to truth from beginning to end, while preserving the deep mainstream of the model to be able to process such blurry images.
- **Merging layers to adapt multiple frames.** The SR layer consists of several convolutional layers and a specially designed merging layer, which is the sole revenue of communication across images. As mentioned above, these images are processed individually thouroughout the network, and they are inputted from the last layer independently, convolutioned independently, and passed to the next layer independently. This apparently cannot fully utilize the information across the multiple frames, so that we design a merging process inside each SR layer, which merges featuremaps from all the images into one, and this data is distributed to each image in this layer and stacked with their featuremaps for the next convolution. In this way, during the refinement process of each of the images in every SR layer, the model has access to data of consensuses among other contemporary images,

which can inspire the network to extract more prominent and collective features, and induce the convergence of all input images for increased seamlessness at the successing merging layer. In most deep learning architectures, the features extracted in each layer is more complex than the previous ones, the former's discoveries built on what the latter has achieved, and our model is not an exception; However, the merging layers we designed can support this increase. At each layer the image is exposed to the consensuses of features from other images at the same level of complexity, acting as a verification for the hypothetical features extracted from the current image, so that in the training process more audacious features can be learned and proposed without fear of punishment from the final loss, increasing the quality of featuremaps throughout the network.

- **Model Complexity and adaption for variable input.** All the convolutional layers throughout the network are thus designed: They are performed individually for each image(or the featuremaps of them from the previous layer), and these convolutions in the same layer share a same set of parameters in training and production, thus reducing parameter count and avoiding convolutions for large numbers of channels, saving calculation power. This also leads to the benefit that the output featuremaps of all the images are the evaluations over the same set of features CL: (???), so that the merging process can also be accomplished easily without the need of trainable parameters. In our design, the merging layers consists of simple pixelwize processes resembling average, min and max processors, to filter out either the collective or the most prominent features among the featuremaps. This process functions with any number of input images. As the convolutional processes are also performed individually for each layer, the same set of parameters of the network model can function normally with any number of input images, while the calculation complexity also increases linearly with the amount of input. On one hand, this gives our model the ability to adapt to the uncertainty of the number of avaliable images, providing relatively satisfactory results regardless whether the attacker has ample time photographing the target phone before it changes its display; on the other hand, the relative independence and the isotropy of the merging layers avoids the reliance on sequential order and consistency between neighboring frames(which is common among most multi-frame SR networks), solving the inconsistency problem mentioned in Figure 1.
- **Compatibility with complex characters.** The discrete distribution of characters leads to the inclination of diviation and producing 'fake' results. The frequent introduction of input images is designed to mitigate this challenge. Also, our model is trained on these characters—the images we collected for the training dataset are all cropped so that only the characters remain, so that during the training processes the model can learn correlations between certain features and strokes, and the regular patterns of the characters, narrowing the possibilities offered by the blurry images. We also introduce adaptive boosting [8] in our training process, increasing the loss weight of the wrongly reconstructed characters, determined by loss in earlier stages and OCR in later stages, to accommodate the imbalanceness of data. The loss functions in training is also redesigned. The mean square error(MSE) is not suitable in this application, as a misplaced stroke, although highly obstructing readability, may only invoke a slight decrease in MSE because it only influenced a few pixels. Our solution is putting a weight on each pixel before applying weighted MSE. Assuming the characters are black on white, this weight increases at darker pixels and propagates to neighbouring dark pixels so that long strokes and intersections of strokes are given higher weights. This process serves as a supplement to MSE to focus more on readability and is beneficial for OCR and human reading tests.

B. Network Design

The core of the SRPeek system is a specially designed multi-frame SR neural network, accepting a group of N images indexed $x_1^{(0)}$ to $x_N^{(0)}$ as input and generating an image with higher resolution y as output. The network comprises L layers, each of which implements a non-linear transformation $H_l(\cdot)$, where l indexes the layer. As mentioned before, in each layer images are processed separately, with the merging layers as a revenue for communication, so that the output of each layer is correspondent to the input. We denote the output of the l^{th} layer as $x_1^{(l)}$ to $x_N^{(l)}$, which is also the input of the $(l+1)^{th}$ layer. Till now the model is not different from traditional SR models:

$$x_i^{(l)} = H_l(x_i^{(l-1)}), i = 1, 2, ...N \tag{1}$$

CL: (function above cannot express merging between x1 x2 ... xN?)

Additionally, we introduce the initial images $x_i^{(0)}$ as an input for all the layers:

$$x_i^{(l)} = H_l(x_i^{(l-1)}, x_i^{(0)}), i = 1, 2, ...N \tag{2}$$

The last layer is exceptional, it yields a single image y as output.

Presently, in these layers nothing is done to raise the resolution of the images, so that the resolution of $x_i^{(0)}$ to $x_i^{(l-1)}$ and y remains the same. To increase resolution we insert several $2\times$ nearest upsampling layers U evenly throughout the architecture between the layers:

$$x_i^{(l)} \leftarrow U(x_i^{(l)}), x_i^{(0)} \leftarrow U(x_i^{(0)}), i = 1, 2, ...N \tag{3}$$

we upsample the input images $x_i^{(0)}$ simultaneously to keep the two inputs of the following layers $H_l(x_i^{(l-1)}, x_i^{(0)})$ unanimous in resolution. For example, in a $4\times$ SR network with five layers, we may insert 2 $2\times$ nearest upsampling layers behind the 2nd and 4th layer.

Inside each layer H_l there are 3 convolution layers $Conv_{1l}(\cdot)$, $Conv_{2l}(\cdot)$, $Conv_{3l}(\cdot)$ and 1 merging layer $Merge_l(\cdot)$.

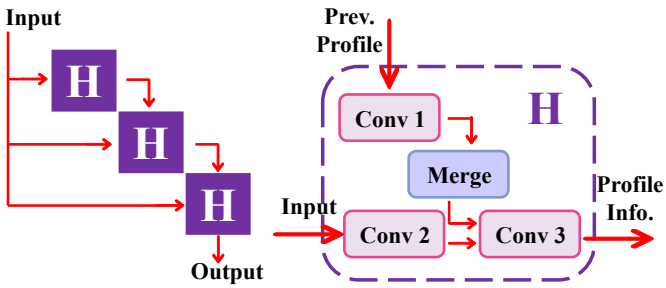


Fig. 3: Network Architecture of SRPeek.

CL: ($H(\cdot)$ or $H(\cdot,\cdot)$?)

Conv1: The first convolutional layer, $Conv1$, accepts the layer's first input parameter $x_i^{(l-1)}$ as input. Note that all three convolutional layers accept a single image(or its featuremaps from the last convolutional layer) as input, the convolutional process is repeated for all the images, and calculations within the same convolutional layer share the same group of parameters all the time (parameters denoted as $Param_{sl}$ for convolutional layer $Conv_{sl}, s = 1, 2, 3$):

$$a_i^{(l)} = Conv_{1l}(x_i^{(l-1)}, Param_{1l}), i = 1, 2, ...N \tag{4}$$

Merge: The results of the previous step of all the images $\{a_1^{(l)}, a_2^{(l)}, ..., a_n^{(l)}\}$ are then passed to the merging layer $Merge_l$ to generate t groups of featuremaps. Suppose the results of $Conv_{1l}$ consists of R channels:

$$a_i^{(l)} = \{a_{i1}^{(l)}, a_{i2}^{(l)}, ..., a_{iR}^{(l)}\}, i = 1, 2, ...N \tag{5}$$

The data in each channel will be merged separately in the merging layer. The output is $T \times R$ channels, denoted as $b_{tr}^{(l)}(t = 1, 2, ...T, r = 1, 2, ...R)$:

$$b_{tr(p,q)}^{(l)} = \sum_{i=1}^{N} a_{ir(p,q)}^{(l)} e^{k_t a_{ir(p,q)}^{(l)}} / \sum e^{k_t a_{ir(p,q)}^{(l)}} \tag{6}$$

where (p,q) represent the pixel at this coordinate, and k_t is a set of fixed parameters shared in all the merging layers throughout the model, controlling the behavior of the merging process. Apparently, k=0 leads to averaging, k=$+\infty$ leads to max operator and k=$-\infty$ leads to min operator. We use T=5 and k=-1,-0.5,0,0.5,1 in our model, giving consideration to both consensuses (k=0,averaging) and prominent features(k=1, 'soft'max and k=-1, 'soft'min). these $T \times R$ channels $b_{tr}^{(l)}$ is the output of this merging layer $Merge_l$.

Conv2: $Conv2$ is a replica of $Conv1$, processing the layer's second input parameter $x_i^{(0)}$, also generating N outputs with R channels per output, denoted as $c_i r^{(l)}, i = 1, 2, ...N, r = 1, 2, ...R$:

$$\begin{aligned} c_i^{(l)} &= Conv_{2l}(x_i^{(0)}, Param_{2l}), i = 1, 2, ...N \\ c_i^{(l)} &= \{c_{i1}^{(l)}, c_{i2}^{(l)}, ..., c_{iR}^{(l)}\}, i = 1, 2, ...N \end{aligned} \tag{7}$$

Conv3: The data from $Merge$ and $Conv2$ are merged together, in that all the $T \times R$ channels of $b_{tr}^{(l)}$ are replicated N times and stacked with each one of the N outputs of $Conv_{2l}$, before these N outputs, each with $(T + 1) \times R$ channels, are passed through the third convolutional layer $Conv_{3l}$. There are also N output of this convolutional layer, denoted as $d_i^{(l)}, i = 1, 2, ...N$:

$$d_i^{(l)} = Conv_{3l}(Stack(c_i^{(l)}, b^{(l)}), Param_{3l}), i = 1, 2, ...N \tag{8}$$

Output: If $l < L$, this is not the last layer, the N outputs of step (4) will be the output of layer H_l. Otherwise, as we need to present a single image y as output, we add another merging and a common convolutional layer after $Conv_{3l}$. The merging layer is identical to the previous $Merge_l$, merging the N outputs $d_i^{(l)}$ into $T \times R$ channels $e_{tr}^{(l)}$, and a convolutional layer processes this $e_{tr}^{(l)}$ to generate a single channel of output y.

$$\begin{aligned} \{e_{tr}^{(L)}, t \le T, r \le R\} &= Merge'_L(\{d_i^{(L)}, i \le N\}) \\ y &= Conv'_L(\{e_{tr}^{(L)}, t \le T, r \le R\}) \end{aligned} \tag{9}$$

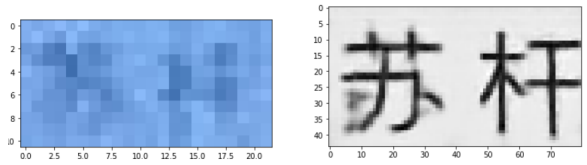
The full structure of a 3 layer network is shown in Figure 3.

V. Implementation & Training

A. Data Collection

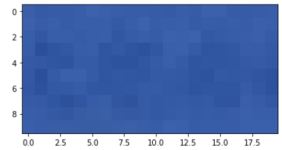
In our experiment, we are forced to collect the training dataset on our own instead of using public datasets. Because of our unique application, to the best of our knowledge there is no publicly available image dataset built for shoulder-surfing. Modern network architectures work with naturally obtained images, e.g. ordinary photos, satellite images, scanned documents, YouTube videos, etc. where the images are well focused and captured within the imaging ability of the camera, with the objects of interest, e.g. the face or the printed word, at least visible to the naked eye. and we apply the SR algorithms to reduce noise and refine details, like repainting texture and refining edges. However, as emphasized above, in our application we face images that is extremely blurred and distorted, due to the extreme circumstances when the photos are taken, and it is impossible to tell from each single picture whether a stroke of a character really belongs here, as it's equally possible that this line is a distorted mixture of multiple lines or is a mutilated part of a longer stroke. These are the reasons why our work does not choose datasets that are publicly available, and collect data by ourselves–as it will completely defeat the purpose if we turn to these datasets. And this also explains why we failed in trying to get comparable results with other SR architectures we use as baselines, as these networks are designed for another purpose and they hardly ever faced such distorted and blurred images.

The task of data collection is quite tedious, as we discovered that we need photos taken in all kinds of environments to train a robust network. We initially collect 60,000 images with the position of the smartphones static and all other environmental parameters stationary, and divided it into training and testing

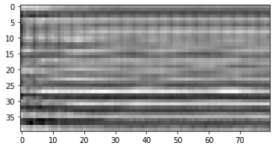


(a) input from same environment

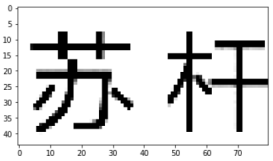
(b) output of (a)



(c) input from different environment



(d) output of (c)



(e) ground truth

Fig. 4: Results of training with data collected in fixed environment. The network performs well in this environment but fails in other.

datasets which do not have common characters between them. Note that different lenses have different photographing abilities and blurring patterns, so the model has to be trained for each phone model. When using smartphones with optical zoom, the images will shift slightly time after time even if we keep them completely still, so the aligning phases are still needed (this shifting is also beneficial in that it simulates normal hand tremors in real-life scenarios even if we fix the phone to a stand instead of holding it up throughout the data collection phase which can last several hours). There are no consistency between adjacent images, so we also shuffle the images to increase this shifting effect and increase the variety of the training data.

Our network easily learned the patterns and presented equally satisfactory results in the test dataset (see Figure 4(b)), displaying clean, easy to read results, showing that the network has learned to extract features beneath the character level, and it is highly possible that this network can recover all kinds of characters, not limited to the characters in the training dataset. However, when we slightly modified the environment, the images displaying the text with a different shade and size, none of the features were successfully extracted, and the network outputted white noise (see Figure 4(d)).

B. Experiment Setting

As a result of these observations, we have to collect images in varied environments. Our experiment consists of 2 smartphones, one for the attacker and one the victim. Their distance is between 1 and 2 meters for traditional lenses(for the camera of the attacker's phone) without optical zooming, and 5 to 7.5 meters for lenses with optical zooming. Both phones are fixed to stands to keep them completely still, but as mentioned above, the lenses with optical zoom will shift slightly time to time, which is similar to a handheld situation. An app runs inside the attacker's phone, taking photos continuously, adjusting its focus and aperture, trying it's best to capture high-quality photos. Another app runs inside the victim's phone, displaying random characters with several fonts and colors, for the attacker to capture. The characters are selected from commonly used Chinese characters with 5 to 10 strokes and the English alphabet, and we divide this assemble into training and testing subsets. As the English characters are apparently easier to classify and reconstruct for the networks, the experiments testing the performance of the network will be performed only on Chinese characters, but it is our observation that our system works on English characters as well as Chinese characters.

As the experiment goes on, the attacker phone will keep on taking photos in burst mode (20 photos per time), and the victim phone will change the characters it is displaying at a fixed interval, when approximately 100 images were taken by the attacker. We change the environmental setting whenever about 2k photos were taken, relocating the phones (while keeping their distance between 1 and 2 meters) or modifying the angle of the phones. The attacker will always point its phone directly at the victim, keeping its screen in the middle of the photos; The victim, however, may tilt its phone at an angle within 30 degrees. We then readjust the focus and aperture of the attacker's phone to maximize the image quality before continuing data collection. Screenshots were taken in the victim's phone each time it changes its display, and these screenshots are scaled, spun and deformed according to the distance and angle between the phones at the time, and used as ground truth for training and evaluation.

The data was collected indoors, as is often the case of shoulder-surfing in public areas (theaters, subways, offices, etc.). We perform the experiment in a room with a window and repeat the data collection phase in different times of the day and night, with curtains drawn or open, lights turned on and off to modify the illumination parameter. The position of the phones is also a crucial factor in this parameter, and we avoid extreme settings, e.g. having the sunlight directly shining on the screen. We collected 800,000 images in this way with a Redmi6A phone (which possess a camera with 130 million pixels). This process is extremely time-consuming, but the data amount is crucial in our experiment, as slight variations of the environment will cause drastic changes in the features extracted from the characters, and only by covering the variations in each of the environmental parameters in the training data can the system successfully function in different scenarios.

C. Model Specifications

It is our observation that cameras in different phones display different patterns of distortion when performing the shoulder-surfing experiment, and apparently, images captured from a phone with weaker abilities (less pixels, less range of focus,

worn lenses, etc.) will need a stronger, more complex model to extract the features. Thus, the specifications of our model are only for reference and may not work when reproduced on another phone.

Our model accepts 20 images at a time, with a size of 9x9 pixels. The model consists of 5 blocks described in the previous section. In each block, feature maps from the previous layer are passed through a single convolutional layer with 32 channels of feature maps as output. the 20x32 feature maps are then merged horizontally with the max-min-average process, and the output is 5x32 channels. Simultaneously, the original input images are processed again with a single convolutional layer with 32 channels output, and stacked with the former 5x32 channels to form a dataflow of 6x32 channels for each one of the 20 images. These channels are finally passed through a single convolutional layer, outputting 32 channels per image for the next block. The kernels in each convolutional layer is 3x3. We also insert LeakyReLU and Batch Normalization processes after each convolutional layer, and 2 upsampling layers between the 5 blocks. A single 1x1 convolutional layer is placed after the 5 blocks with a single channel as output to form the final output layer. This model consists of approximately 200,000 parameters and a complexity of about 400,000 FLOPs. As a light-weighted model, it makes a prediction within 0.1 seconds on a Tesla K80 GPU over a 9x9 patch, and when implemented on a smartphone, the human user can recognize a character within 2 seconds of processing time.

D. Training Process

Because of the difficulties in discovering the patterns among the blurry images, the training process is not so straightforward and somewhat time-consuming. Our approach is as follows: we first collect 60,000 images at a stationary experiment setting and train the model with it, as mentioned in Sec V-A, until we get satisfactory results on the test dataset. The model should be able to handle this task easily. After that, we use transfer learning methods to fine tune the model in order to fit different environmental parameters. We add a small percentage of image data from another similar experiment setting into the training data and resume training. When the model stabilizes, continue adding data from the same setting until the model can equally process data from the two image collections. Note that in this process the model will tend to extract false features, displaying wrong but clear texts as result. This phenomenon can be mitigated with dropout and normalization layers. As the model successfully fits two of the image sets, we repeat this procedure for several times until it shows signs of self-adapting, for example, when presented photos taken at 2.0m and 1.5m range in the training process, the model fits 1.75m photos easily. After that, the model can learn with the full fully-shuffled dataset with fewer difficulties.

VI. Evaluation & Case Study

We evaluate the ability of the network model, testing its performance in different training and testing conditions. We perform the following experiments with two phones: a Redmi 6A smartphone, with a single rear camera with 13 million pixels and digital zoom only, and a HUAWEI P40 Pro, with multiple rear cameras, the telephoto camera(which we use in the experiments) possessing up to 5x optical zooming ability which we will utilize fully in our experiments.

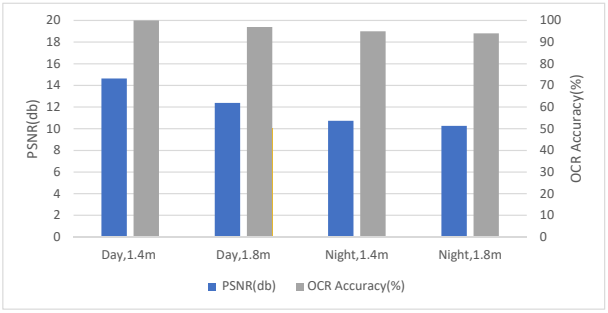


Fig. 5: Performance in Controlled Environment(Traditional Lenses)

A. Performance In Controlled Environment

We train and test the model with the images captured with exactly the same environment parameters, and used Peak Signal to Noise Ratio to evaluate the accuracy of the recovered images(see Figure 5 and Figure 6). Moreover, the ultimate goal of our system is the readability of the recovered images, so we also used Optical Character Recognition(OCR) services to evaluate the accuracy. The model with traditional lenses(without optical zoom) is trained and tested at 1 2 meters range, while the one with optical zoom at 5 7.5 meters, at which distance less than 5% of the characters(only the simplist ones) can be recognized by humans from the photos without the assistance of SR algorithms. The model can achieve an accuracy above 90% at 1.8m with traditional lenses and 6m with optical zooming lenses. Considering the complexity of Chinese characters, and the assist of context when read by an OCR recognizer, we believe this level of accuracy can provide sufficient data at a shoulder surfing scenario, thus proving the efficiency of our model. In the optical zoom group the accuracy of the model dropped drastically at 7m distance. Increased distances means less data and less restrictions of the possible outputs, which leads to artifacts(missing or misplaced strokes,etc.). It's the nature of Chinese characters that one mistaken stroke will largely affect its readability, leading to the result that while the pixelwize error rises steadily with the increased distance, the accuracy will experience a drastic drop.

B. Performance In Random Environment

We train the model with data captured in different environments, as mentioned before, and test its ability in other

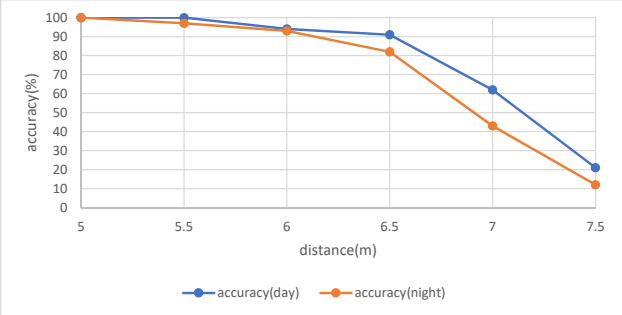


Fig. 6: Performance in Controlled Environment(Optical Zoom Lenses)

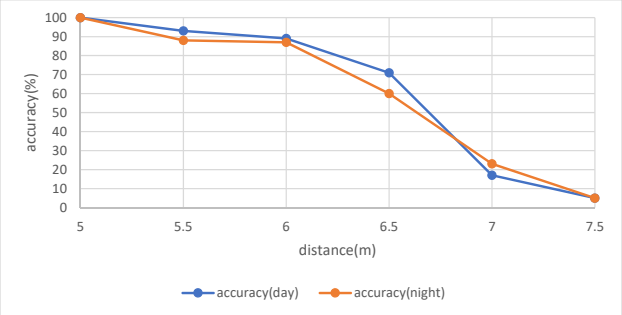


Fig. 8: Performance in Random Environment(Optical Zoom Lenses)

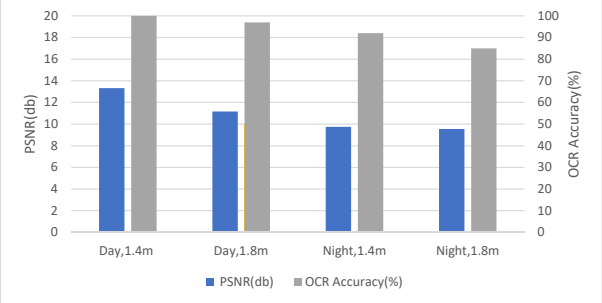


Fig. 7: Performance in Random Environment(Traditional Lenses)

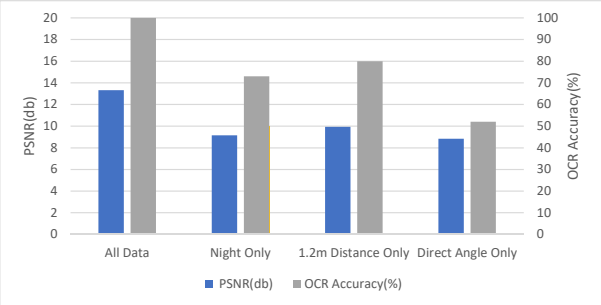


Fig. 9: Adapting Ability(Traditional Lenses)

environments(see Figure 7 and Figure 8). The model achieves an accuracy above 90% at 1.8m with traditional lenses and 6m with optical zooming lenses.

We compare the results to discover the influence of the surrounding environment: The model performs best in daytime and weaker illumination may decrease its ability. Also, the model performs better at closer distances. As the model is trained initially on the 5m group data, it performs expecially well at this distance, while the accuracy at other distances experience certain levels of decrease. Also, similar to the experiments at controlled environments, random artifacts start to encompass OCR compensation, rendering the result images highly unreadable, until at 7.5m distance only the simplist characters can escape from being flooded by noize.

C. Adapting Ability

We train the model with fewer groups of data, exposing it to fewer variations of environment parameters, and examine the model's performance in other environments. The results are shown in Figure 9.Only the results from the traditional lenses is shown, as the optical zoom lenses group face longer distances and more complex distortions, and with incomplete data the model fails to manage any reasonable reconstruction on the test dataset, resulting in extremely low accuracy in all experiments.

We observed that variations in light and angle parameter in training data is crucial to a robust model. Variations in distance is not so influential to the results, given that the distances are between 1 and 2 meters. Distance changes do not have such a large impact on the size of the characters shown in the images, so that features extracted from a fixed distance might still exist when these characters vary slightly in size. However, if the network is not exposed to angled images during the training process, the rotations and deformations caused by these angles will easily disturb the feature extraction process.

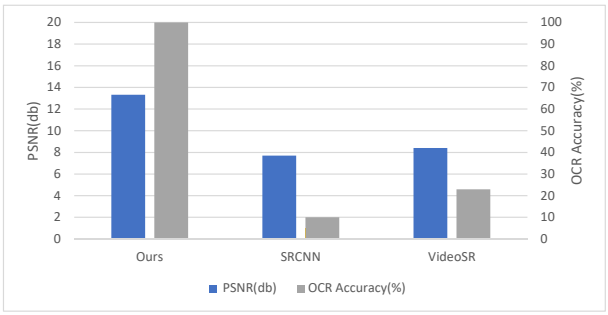


Fig. 10: Comparison with other architectures

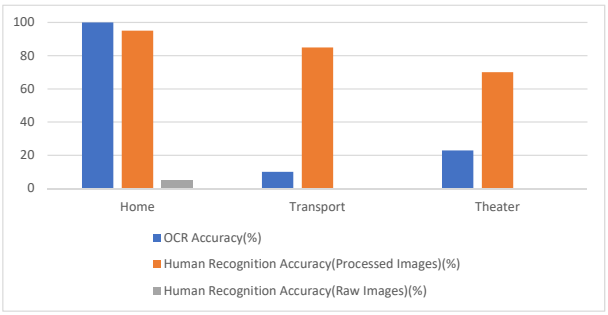


Fig. 11: Accuracy in different real-life scenarios

D. Comparison with other architectures

We train and test other commonly used architectures with the same sets of data and evaluate their results. We chose SRCNN, a commonly used single image SR network, and applied it to each single image before merging the results by pixel-level averageing. We also used a multi-frame version of CNN consisting of 3D convolutional layers, designed for video super resolution(VideoSR). However, as mentioned above, it is very difficult for the single image approaches to utilize information and distinguish the noized and deformed patterns, while VideoSR approaches rely apon consistency between frames, so they fail to give satisfactory results. We used the relatively easy 'daytime 1.2m-distance direct with traditional lenses' group of data for testing. The results are shown in Figure 10.

E. Accuracy

We build the system on smartphone and evaluate its performance in real-life environments. We experiment with a Redmi 6A smartphone (with a camera of 13 million pixels) for the attacker and a HUAWEI Mate8 smartphone for the victim. As the telephoto cameras can assist the attacker to see clearly at approximately 3m distance without any aid from SR algorithms, we believe it's insignificant to further extend this distance to judge it as a threat to privacy, so that the following experiments are performed with traditional lenses at 1 2m range. We ask 5 human participants to read the reconstructed characters to evaluate the usability of our model. No participants can read the unprocessed images, but all of them can decipher the information on the reconstructed image without much difficulty. The results are shown in Figure 11.

F. Influence of hand tremors

We ask 5 participants to capture images with handheld smartphones, keeping their hand still to their greatest effort(Handheld camera). We process these images and let them read the results. We compare this performance to the data collected on stationary phones to evaluate the influence of hand tremors. We also ask participants to hold a smartphone in their hands and read a piece of text, without other additional instructions(Handheld target). The user may freely interact with the phone when reading. We capture images of the phone at the same time to see how our system deal with a moving target screen.The results are shown in Figure 12.

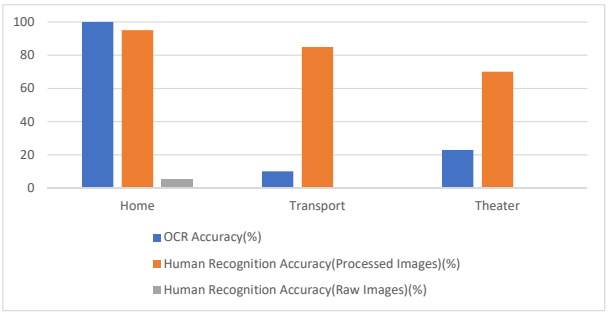


Fig. 12: Influence of hand tremors

We conclude from the results that hand tremors can impact the performance of our system. Although the edges of the phone are notable marks for image alignment, small shifts in the sub-pixel level will cause blurriness in the results of the networks, lowering the readability of the outputs.

G. Success rate in different tasks

We test the success rate of obtaining crucial information when the observed participant perform several tasks on a phone: reading text message, typing text message, entering PIN, and typing password with numbers, English and special characters. The observed participant will turn off the screen of his phone as soon as he/she finishes the task, while the

Scenario	Read text	Type text	Enter PIN	Enter password
Human Recognition Accuracy (raw image)	5	0	0	0
Accuracy	100	100	-	-
Accuracy per character	-	-	100	80

TABLE II: Success rate in different tasks

observing participant will observe through our APP, constantly capturing images and processing them to display a real-time and magnified view of the observed phone. For the first two tasks,we ask the observing participant several questions to test if he/she have collected the vital information(e.g. the name or the location mentioned in the text). For the task of recognizing PIN and passwords, we use accuracy per character as a supplementary evaluation metric. In these two scenarios, we ignore the virtual keyboard and view only the textbox, assuming that the last character of the entered string is always visible. Less photos will be avaliable for each character but decifering English characters is also easier than Chinese characters, and we trained a model specifically for each task(with the same nerual network architecture), the training images containing only English characters and numbers(or only numbers for the PIN entry scenario). The results are shown in Table II.

We prove from this experiment that our system functions normally in everyday scenarios and poses a threat to screen privacy.

H. Perceived shoulder surfing susceptibility

We ask the observed participant to rate the perceived shoulder-surfing susceptibility in these scenario. The attacker will be sitting or standing behind the participant at 1.5m range, pretending to be interacting with their own phone while continuously running the shoulder-surfing APP. None of the participants were alerted by the attacker's behavior and reported suspicion of shoulder-surfing. We believe that our system can enable a malicious attacker to gather large amounts of critical information from the victim while remaining unnoticed.

VII. Limitations & Discussions

Image Capturing Ability Latest models of smartphones can capture images at 10 frames per second in burst mode easily, however, this ability is not common in phones that are 3 years old. Heavily used phones also performs less than ideal when capturing images in burst mode. Also, when capturing images the user need to hold their phone as still as possible to avoid motion blur and assist image alignment. The user also has to keep a sharp focus on the target phone.

Processing Ability To achieve best performance the user needs a phone with strong processing capabilities to run the neural network at real time. As neural networks have been common place in numerous modern APPs, most phones of the latest generation have upgraded their processing ability to run neural networks, but older versions might not possess such processing powers and cannot process images at real-time.

Motion and Tremors In our experiment we assume the observed user will hold still his/hers phone, and not making interactions too often. However some users may tilt their phones during interactions—especially when typing with one hand. Too much tremor will cause motion blur in captured images and misalignment between frames, thus degrading the result.

Although our shoulder surfing system proves to be highly efficient against unprotected screens, there are some simple methods to mitigate this unique threat while not cumbering the user.

Dynamic background. Most multi-frame SR algorithms are designed based on the assumption that all input images are reflections of the same scene, and ours is not an exception. By deploying a dynamic background behind the characters, such as tiny dots and lines travelling slowly around the screen, we can construct a constantly changing scene that will confuse the multi-frame SR algorithms, and due to the blurriness of the images, these influential elements cannot be easily removed. These dots do not need to be distinct or colored same as the the texts, as multi-frame SR algorithms function with tiny,pixel level differences between frames, making them especially sensitive to microscopic changes.

Active scanning. There are several works providing an active countermeasure against shoulder surfing threats with the naked eye. With front-facing cameras and face detection algorithms, the smartphone can constantly scan the surrounding passers-by and detect their gaze direction, and give a warning to the user when that gaze points at the screen. However, to the extent of our knowledge none of these works have included cameras into their detection scope, but we believe it's practical to implement such features.

Adversarial machine learning methods. In recent years we have discovered the weaknesses of neural networks and that inserting certain microscopic changes, undetectable to the human eye, to the pixels of an image will make it look different to a neural network. These methods fools the feature extraction phases of neural networks, so that SRPeek is also vulnerable to this attack. Theoretically, by exerting a pattern to the victim's screen, it can be captured by the attacker's camera and confuse its SR algorithms, but these remains to be implemented in our future work.

VIII. Conclusion

In this work we designed a holistic system, SRPeek, for shoulder surfing on smartphone, which serves as an up-to-date version of a threat model for shoulder surfing, and proved its efficiency. We proved that this threat towards screen privacy is imminent and can steal critical information, including personal texts or passwords, from long distances, thus escaping detection. It is our wish that this work can stur discussion in

the field of screen privacy protection and proporgate defense mechanisms across critical mobile apps.

The core of SRPeek is a specially designed multi-frame SR network. With its innovative architecture this network outperforms other algorithms of the same field in our application. The design ideology enables this network to process higher levels of data integration ability while keeping a low calculation profile, and we believe the elements of this design can be used in other applications with large amounts of data, such as natural language processing or anomaly detection. Our model can also be used in OCR tasks when multiple images are avaliable, functioning as a preprocessor to improve the quality of the images and increase accuracy.

REFERENCES

[1] F. Brudy, D. Ledo, and S. Greenberg. Is anyone looking? mediating shoulder surfing on public displays (the video). In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 159–160. 2014.

[2] N. Chakraborty and S. Mondal. Tag digit based honeypot to detect shoulder surfing attack. In *International Symposium on Security in Computing and Communication*, pages 101–110. Springer, 2014.

[3] C.-Y. D. Chen, B.-Y. Lin, J. Wang, and K. G. Shin. Keep others from peeking at your mobile device screen! In *The 25th Annual International Conference*, 2019.

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[5] Z. Dong, S. Zhang, B. Ma, D. Qi, L. Luo, and M. Zhou. A hybrid multi-frame super-resolution algorithm using multi-channel memristive pulse coupled neural network and sparse coding. In *2019 7th International Conference on Information, Communication and Networks (ICICN)*, pages 185–190, 2019.

[6] M. Eiband, M. Khamis, E. Von Zezschwitz, H. Hussmann, and F. Alt. Understanding shoulder surfing in the wild: Stories from users and observers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 4254–4265, 2017.

[7] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Robust shift and add approach to superresolution. In *Applications of Digital Image Processing XXVI*, volume 5203, pages 121–130. International Society for Optics and Photonics, 2003.

[8] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, page

[15] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

148–156, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[9] W. Goucher. Look behind you: the dangers of shoulder surfing. *Computer Fraud & Security*, 2011(11):17–20, 2011.

[10] Y. Huang, W. Wang, and L. Wang. Video super-resolution via bidirectional recurrent convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):1015–1028, 2017.

[11] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016.

[12] M. Kawulok, P. Benecki, S. Piechaczek, K. Hrynczenko, D. Kostrzewa, and J. Nalepa. Deep learning for multiple-image super-resolution. *IEEE Geoscience and Remote Sensing Letters*, 2019.

[13] M. Kumar, T. Garfinkel, D. Boneh, and T. Winograd. Reducing shoulder-surfing by using gaze-based password entry. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 13–19, 2007.

[14] T. Kwon, S. Shin, and S. Na. Covert attentional shoulder surfing: Human adversaries are more powerful than expected. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(6):716–727, 2013.

[16] J. Lyn and S. Yan. Image super-resolution reconstruction based on attention mechanism and feature fusion. *arXiv preprint arXiv:2004.03939*, 2020.

[17] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero. Poster: Fast, automatic iphone shoulder surfing. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 805–808, 2011.

[18] H. Nasrollahi, K. Farajzadeh, V. Hosseini, E. Zarezadeh, and M. Abdollahzadeh. Deep artifact-free residual network for single-image super-resolution. *Signal, Image and Video Processing*, 14(2):407–415, 2020.

[19] A. Papadopoulos, T. Nguyen, E. Durmus, and N. Memon. Illusionpin: Shoulder-surfing resistant authentication using hybrid images. *IEEE Transactions on Information Forensics and Security*, 12(12):2875–2889, 2017.

[20] Polotiko. Aguirre furious at photo leak of private text message, 2017.

[21] A. Saad, M. Chukwu, and S. Schneegass. Communicating shoulder surfing attacks to users. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia*, pages 147–152, 2018.

[22] M. S. Sajjadi, R. Vemulapalli, and M. Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6626–6634, 2018.

[23] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[24] N. Suetake, M. Sakano, and E. Uchino. Image super-resolution based on local self-similarity. *Optical review*, 15(1):26–30, 2008.

[25] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget. Design and evaluation of a shoulder-surfing resistant graphical password scheme. In *Proceedings of the working conference on Advanced visual interfaces*, pages 177–184, 2006.